

A Proxy-Based Dynamic Inheritance of Soft-Device

Jia Bi^{1,2,3}, Yanyan Li^{1,2}, Yunpeng Xing^{1,2}, Xiang Li^{1,2}, and Xue Chen^{1,2}

¹ Institute of Computing Technology, Chinese Academy of Sciences, China
{bj, liyy, ypxing}@kg.ict.ac.cn

² Graduate School of Chinese Academy of Sciences

³ Oracle, China Development Center, China

Abstract. Soft-device is a promising infrastructure of next-generation distributed system. Soft-devices are configurable and adaptive software virtual mechanism, providing services to each other and to other virtual roles according to the content of the resources and related configuration information. In order to aggregate and reuse the existing soft-devices for more complex applications, this paper proposes a dynamic inheritance mechanism of soft-devices. By taking into account the novel characteristics of soft-devices, this approach provides flexible and effective inheritance modes, which overcomes the limitation of traditional local inheritance. It will play an important role in future interconnection environment.

1 Introduction

Inheritance mechanism has been investigated in software engineering and artificial intelligence [1, 2, 3, 4, 5]. The Knowledge Grid is an intelligent sustainable internet application environment that enables people or virtual roles to effectively capture, publish, share, and manage explicit knowledge resources [6, 8, 9].

Soft-devices are configurable and adaptive software virtual mechanism, representing distributed network software and devices. It takes the advantages of the active and intelligent features of the intelligent agents, the semantic-based features of the semantic web, the advantages of the Knowledge Grid, and the configurable feature of general-purpose computers [7, 10, 12, 13]. This paper implements soft-device inheritance.

Based on the notion and mechanism introduced in [7, 11], a soft-device comprises six components, which can be defined as a six tuple:

$$SD = \langle C, K, D, W, E, I \rangle$$

Where C, K, D, W, E, I respectively denotes the container, knowledge base, detector, built-in workflows, explainer, and interface.

2 Inheritance Mechanism of Soft-Device

2.1 Dynamic Single Inheritance

Let $Loc1 = Loc3 \prec Loc2$ and ΔSD is unchangeable. The definition of the dynamic inheritance becomes:

$$SD1(t, Loc1) = SD2(t, Loc2) \oplus \Delta SD(Loc1)$$

This type of inheritance is named as dynamic single inheritance. Because each child soft-device inherit only one parent soft-device, there are only one proxy resource in the soft-device, namely $Np2=1$. Because ΔSD is local increment, there are no proxy resource in it, namely $Np3=0$. The corresponding operations of components are defined as followings:

$$\begin{aligned}
 SD1(t, Loc1).C &= SD2(t, Loc2).C \oplus \Delta SD(Loc1).C \\
 &= \langle \{Rec_{2_1}(t, Loc2), \dots, Rec_{2_{Nr_2}}(t, Loc2), \Delta Rec_{c_1}(Loc1), \dots, \Delta Rec_{c_{Nr_3}}(Loc1)\}, \{Pro_{2_1}(t, Loc2)\} \rangle \\
 SD1(t, Loc1).K &= SD2(t, Loc2).K \oplus \Delta SD(Loc1).K \\
 &= \{Rul_{2_1}(t, Loc2), \dots, Rul_{2_{Nr_2}}(t, Loc2), \Delta Rul_1(Loc1), \dots, \Delta Rul_{Nr_3}(Loc1)\} \\
 SD1(t, Loc1).D &= SD2(t, Loc2).D \oplus \Delta SD(Loc1).D = SD2(t, Loc2).D \downarrow \Delta SD(Loc1).D = NewD \\
 SD1(t, Loc1).E &= SD2(t, Loc2).E \oplus \Delta SD(Loc1).E = SD2(t, Loc2).E \downarrow \Delta SD(Loc1).E = NewE \\
 SD1(t, Loc1).W &= SD2(t, Loc2).W \oplus \Delta SD(Loc1).W \\
 &= \{wf_{2_1}(t, Loc2), \dots, wf_{2_{Nw_2}}(t, Loc2), \Delta wf_{f_1}(Loc1), \dots, \Delta wf_{Nw_3}(Loc1)\} \\
 SD1(t, Loc1).I &= SD2(t, Loc2).I \oplus \Delta SD(Loc1).I \\
 &= \{Ser_{2_1}(t, Loc2), \dots, Ser_{2_{Ns_2}}(t, Loc2), \Delta Ser_{f_1}(Loc1), \dots, \Delta Ser_{Ns_3}(Loc1)\}
 \end{aligned}$$

Because of the dynamic and local facets, the dynamic single inheritance has not the transfer characteristic compared with traditional inheritance mode.

The operation of single inheritance of soft-devices results in abstraction hierarchies that take the form of tree or more generally directed acyclic graph (DAG). Because of remote inheritance, a soft-device may inherit his child soft-device, which generates circular inheritance. Thus, in order to void circular inheritance, a soft-device can not inherit his child.

We define function *IsOne* to mean whether two soft-devices is the same one. If the result is true, the two soft-devices is exactly the same one, or else they are two soft-devices. It is easy to judge, because soft-devices are registered on a site, and they have the unique id.

$$IsOne(SD1, SD2) = \begin{cases} true \\ false \end{cases}$$

Consider that SD1 will inherit SD2. The following algorithm judge whether there are circular inheritance.

Algorithm

```

If (IsOne(SD1,SD2)= true)
then return there exists circular inheritance
else
{
    Put child soft-devices of SD1 in a Queue.
    while (Queue is not empty)
    {
        Pop a soft-device SD from Queue,
        if (IsOne(SD,SD1)=true )
            then return there exist circular inheritance,
            else Put child soft-devices of SD1 in a Queue.
    }
    Return There are not circular inheritance.
}

```

2.2 Dynamic Multiple Inheritance

Let $Loc1 \langle \rangle Loc2$, $Loc2 \langle \rangle Loc3$, $Loc1 \langle \rangle Loc3$, the inheritance is dynamic multiple inheritance. We can change the definition as follows:

$$SD1(t, Loc1) = SD2(t, Loc2) \oplus SD3(t, Loc3)$$

We can consider SD2 is the parent soft-device and the SD3 is the incremental part, or SD3 is the parent soft-device and SD2 is the incremental part. The operation is the same.

We can easily expand the definition to multiple inheritances.

$$SD5(t, Loc5) = SD1(t, Loc1) \oplus SD4(t, Loc4)$$

$$SD5(t, Loc5) = (SD2(t, Loc2) \oplus SD3(t, Loc3)) \oplus SD4(t, Loc4)$$

Because the \oplus operation has the law of exchange and combine, the equation can be written as follows:

$$SD5(t, Loc5) = SD2(t, Loc2) \oplus SD3(t, Loc3) \oplus SD4(t, Loc4)$$

2.3 Dynamic Characteristic of Inheritance

The parent soft-device change dynamically, and the child soft-device can change correspondingly. This section discusses the machine of realization.

In the moment t , the soft-device satisfies the following equation:

$$cSD(t, Loc1) = pSD(t, Loc2) \oplus \Delta SD(Loc1)$$

In the next moment $t+1$, $pSD(t, loc2) \rightarrow pSD(t+1, loc2)$, namely $pSD(t, loc2) = \langle pC(t, loc2), pK(t, loc2), pD(t, loc2), pE(t, loc2), pW(t, loc2), pI(t, loc2) \rangle \rightarrow pSD(t+1, loc2) = \langle pC(t+1, loc2), pK(t+1, loc2), pD(t+1, loc2), pE(t+1, loc2), pW(t+1, loc2), pI(t+1, loc2) \rangle = \langle pC(t+1, loc2), pK(t+1, loc2), pD(t+1, loc2), pE(t+1, loc2), pW(t+1, loc2), pI(t+1, loc2) \rangle$
 $cSD(t+1, Loc1) = pSD(t+1, Loc2) \oplus \Delta SD(Loc1)$.

According to the definition of \oplus operation, the cSD should be:

$$cSD(t+1, Loc1).C = \{ pRe_{c_1}(t+1, Loc2), \dots, pRe_{c_{Nr_2}}(t+1, Loc2), \Delta Re_{c_1}(Loc1), \dots, \Delta Re_{c_{Nr_3}}(Loc1) \}, \\ \{ Pro_{o_1}(t+1, Loc2) \} >$$

$$cSD(t+1, Loc1).K = \{ pRul_{Nu_1}(t+1, Loc2), \dots, pRul_{Nu_2}(t+1, Loc2), \Delta Rul_1(Loc1), \dots, \Delta Rul_{Nu_3}(Loc1) \}$$

$$cSD(t+1, Loc1).D = NewD(t+1, Loc1)$$

$$cSD(t+1, Loc1).E = NewE(t+1, Loc1)$$

$$cSD(t+1, Loc1).W = \{ pWf_1(t+1, Loc2), \dots, pWf_{Nw_2}(t+1, Loc2), \Delta wf_1(Loc1), \dots, \Delta wf_{Nw_3}(Loc1) \}$$

$$cSD(t+1, Loc1).I = \{ pSer_1(t+1, Loc2), \dots, pSer_{Ns_2}(t+1, Loc2), \Delta Ser_1(Loc1), \dots, \Delta Ser_{Ns_3}(Loc1) \}$$

Because Pro is local mirror of pSD, in the moment $t+1$

$$Pro(t+1, loc1) = pSD(t+1, Loc2)$$

The child soft-device is

$$cSD(t+1) = \langle \langle \Delta SD.C, Pr o(t+1) \rangle, \Delta SD.K, NewD, NewE, \Delta SD.W, \Delta SD.I \rangle$$

This means the parent soft-device change, we only need change the proxy, the other components don't change.

$pSD(t+1, loc2)$ has three status:

- (1)Stop. $pSD(t+1, loc2)$ stops server for others. pSD informs his child soft-devices and disable them.
- (2)Limited Change. The interface component of pSD does not change, namely $pI(t+1, loc2) = pI(t, loc2)$. Because the interface does not change, the proxy of the pSD doesn't change, and his child soft-devices needn't change.
- (3)Change. The interface component of pSD changes, namely $pI(t+1, loc2) \neq pI(t, loc2)$. Its child soft-devices need change correspondingly.

There are three possibilities:

- Servers increase. $pSD(t+1, loc2)$ provides more services, namely $pI(t, loc2) \subset pI(t+1, loc2)$. Servers increasing do not affect the child soft-devices, and the child soft-devices get the new proxy of pSD.
- Servers decrease. $pSD(t+1, loc2)$ provides less services, namely $pI(t+1, loc2) \subset pI(t, loc2)$. Servers decreasing may make the child soft-devices work incorrectly, if the child soft-devices use its servers.
- Servers change. $pSD(t+1, loc2)$ changes some services, namely $(\exists Ser_i \in pI(t+1, loc2) \wedge Ser_i \notin pI(t, loc2)) \wedge (\exists Ser_j \in pI(t, loc2) \wedge Ser_j \notin pI(t+1, loc2))$.

3 Implementation

The architecture of inheritance is shown in the Fig.1.

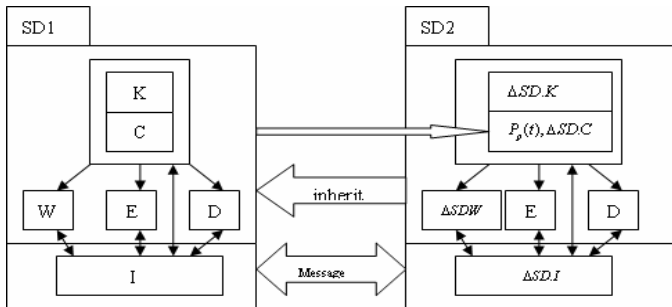


Fig. 1. Architecture of inheritance

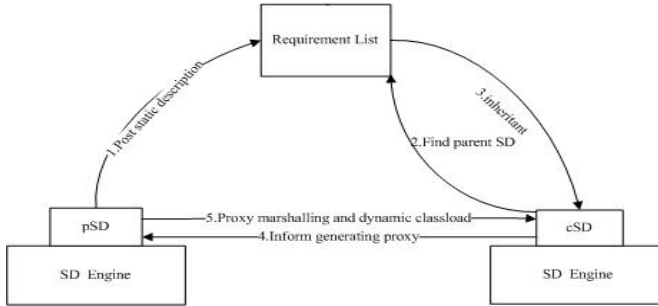


Fig. 2. The process of inheritance

Fig. 2 describes the process of inheritance, where the requirement list plays the same role as in [7].

The process of dynamic inheritance is described in detail as follows:

- Step1. The owner post static description of his soft-device on the requirement list.
- Step2. The other one search the requirement list, and find the soft-device which he requires.
- Step3. According the static description of parent soft-device, add incremental part, and then form the static description of child soft-device.
- Step4. The static description of child soft-device is parsed to form dynamic instance of the child soft-device by the generator. At the same time, parent soft-device and child soft-device establish the message pipe. The process of building dynamic instance of the child soft-device includes the following three steps.
 - Step4.1. The child soft-device informs the parent soft-device generate proxy of its.
 - Step4.2. The parent soft-device marshals the proxy, and sends it to child soft-device.
 - Step4.3. The child soft-device unmarshals the proxy, and dynamic loading in the container component of child soft-device.
 - Step4.4. The each part of constitutes the corresponding component of the child soft-device.

4 Conclusion

Our vision is to provide soft-device as the basis of the future interconnection environment [14, 15]. In order to fit the requirements that soft-device is deployed in the distributed system, this paper proposes the architecture of soft-devices, in which the soft-devices are entirely encapsulated and accessed by interface. Additionally, this paper proposes a proxy-based inheritance mechanism. Different from the traditional inheritance, this mechanism is dynamic and selective, enabling a more flexible and effective inheritance in different scenarios.

References

1. Booch, G. Object Oriented Design with Applications. Redwood City, Calif.: Benjamin/Cummings Pub. Co., 1991.
2. Horty, J.F., Thomason, R.H., and Touretzky, D.S. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. *Artificial Intelligence*, 42 (1990) 311-348.
3. Taivalsaari, A. On the Notion of Inheritance. *ACM Computing Surveys*, 28, 3 (1996) 438-479.
4. Tamma, V.A.M. and Bench-Capon, T.J.M. Supporting Inheritance Mechanisms in Ontology Representation, *EKAW 2000*, LNAI, 1937 (2000) 140-155.
5. Zhuge, H. Inheritance Rules for Flexible Model Retrieval, *Decision Support Systems*, 22 (4) (1998) 383-394.
6. Zhuge, H. A Knowledge Grid Model and Platform for Global Knowledge Sharing, *Expert Systems with Applications*, 22 (4) (2002)313-320.
7. Zhuge, H. Clustering Soft-Devices in Semantic Grid, *IEEE Computing in Science and Engineering*, 4 (6) (2002) 60-62.
8. Zhuge, H. China's E-Science Knowledge Grid Environment, *IEEE Intelligent Systems*, 19(1) (2004) 13-17.
9. Zhuge, H. The Knowledge Grid, *World Scientific Publishing Co.*, 2004.
10. Zhuge, H. Resource Space Grid: Model, Method and Platform, *Concurrency and Computation: Practice and Experience*, 16 (14) (2004) 1385 – 1413.
11. Zhuge, H. Soft-Device Inheritance in the Knowledge Grid, *Springer LNCS*, 3505 (2005) 62-78.
12. Zhuge, H. Semantic Grid: Scientific Issues, Infrastructure, and Methodology, *Communications of the ACM*. 48(4) (2005)117-119.
13. Zhuge, H. and Xing, Y. Integrity Theory for Resource Space Model and Its Application, Keynote at WAIM2005, *Springer LNCS* 3739(2005)8-24.
14. Zhuge, H. The Future Interconnection Environment, *IEEE Computer*, 38(4) (2005) 27-33.
15. Zhuge, H. Exploring an Epidemic in an E-Science Environment, *Communications of the ACM*, 48(9) (2005)109-114.